Copylist()

Vulnerable to TOCTOU issues

Sean Barnum, Cigital, Inc. [vita[1]]

2007-03-20

Part "Original Cigital Coding Rule in XML"

Mime-type: text/xml, size: 6420 bytes

Attack Category	• Path spoofing or confusion problem
Vulnerability Category	• Indeterminate File/Path • TOCTOU - Time of Check, Time of Use
Software Context	• Filename Management
Location	• libgen.h
Description	Care must be exercised when accessing files from passed in filenames. copylist (const char *filenm, off_t *szptr) copies a list of items from a file specified by filenm into freshly allocated memory. The pointer szptr is a variable that contains the integer representing the size of the file that will be stored. There is the possiblity of changes to the filename after copylist is called and thus copy items from an arbitrary file into memory.
APIs	**FunctionName** / **Comments** copylist() /
Method of Attack	The key issue with respect to TOCTOU vulnerabilities is that programs make assumptions about atomicity of actions. It is assumed that checking the state or identity of a targeted resource followed by an action on that resource is all one action. In reality, there is a period of time between the check and the use that allows either an attacker to intentionally or another interleaved process or thread to unintentionally change the state of the targeted resource and yield unexpected and undesired results An attacker could link the passed in filename to another known file after the check to ensure the attacker has permissions to the original targeted file, but before the file is read or written to giving the attacker access to files he should not have access to.
Exception Criteria	If proper checking is performed or user-specified input is not used, this is not a problem.

1. http://buildsecurityin.us-cert.gov/bsi-rules/35-BSI.html (Barnum, Sean)

Solutions

Solution Applicability	Solution Description	Solution Efficacy
Generally applies to copylist()	The most basic advice for TOCTOU vulnerabilities is to not perform a check before the use. This does not resolve the underlying issue of the execution of a function on a resource whose state and identity cannot be assured, but it does help to limit the false sense of security given by the check. Don't perform a check prior to use.	Does not resolve the underlying vulnerability but limits the false sense of security given by the check. Checking the file permissions after the operation does not change the fact that the operation may have been exploited but it does allow halting of the application in an error state to help limit further damage.
When user specification of the file to be altered is not necessary.	Do not rely on user-specified input to determine what path to format.	This will reduce exposure but will not eliminate the problem.
Generally applies to copylist()	Limit the interleaving of operations on files from multiple processes.	Does not eliminate the underlying vulnerability but can help make it more difficult to exploit.
Generally applies to copylist()	Limit the spread of time (cycles) between the check and use of a resource.	Does not eliminate the underlying vulnerability but can help make it more difficult to exploit.

	Generally applies to copylist()	Recheck the resource after the use call to verify that the action was taken appropriately.	Effective in some cases.

Signature Details	

Examples of Incorrect Code	`#include`
	`#include`
	`/* check permissions to the path */`
	`if(!access(file, ...){`
	`/* format path into path token */`
	`copylist(fileName, fileSize);`
	`}`
	`else{`
	`/* permission was denied */`
	`}`

Examples of Corrected Code	
Source References	• ITS4 Source Code Vulnerability Scanning Tool [2] • http://techpubs.sgi.com/library/tpl/cgi-bin/getdoc.cgi?coll=0650&db=man&fname=/usr/share/catman/p_man/cat3g/copylist.z[3] • Viega, John & McGraw, Gary. *Building Secure Software: How to Avoid Security Problems the Right Way*. Boston, MA: Addison-Wesley Professional, 2001, ISBN: 020172152X, ch 9;
Recommended Resources	• M. Bishop and M. Dilger, "Checking for Race Conditions in File Accesses[4]," Technical Report, CSE-95-10, September 1995. • M. Bishop and M. Dilger, "Checking for Race Conditions in File Accesses[5]," *Computing Systems 9* (2) pp. 131-152 (Spring 1996).
Discriminant Set	**Operating System**: • UNIX (All) **Languages**: • C • C++

Cigital, Inc. Copyright

The Build Security In (BSI) portal is sponsored by the U.S. Department of Homeland Security (DHS), National Cyber Security Division. The Software Engineering Institute (SEI) develops and operates BSI. DHS funding supports the publishing of all site content.

1. mailto:copyright@cigital.com